

3

Flujos de ejecución

Grado en Ingeniería Electrónica de Comunicaciones

Luis Hernández Yáñez

Raquel Hervás Ballesteros

Virginia Francisco Gilmartín

Javier Arroyo Gallardo

Facultad de Informática
Universidad Complutense



Índice

Control de flujo

Selección simple

Selección múltiple

Switch

Iteración

While

Do..while

For

Ámbito y visibilidad



Control de flujo

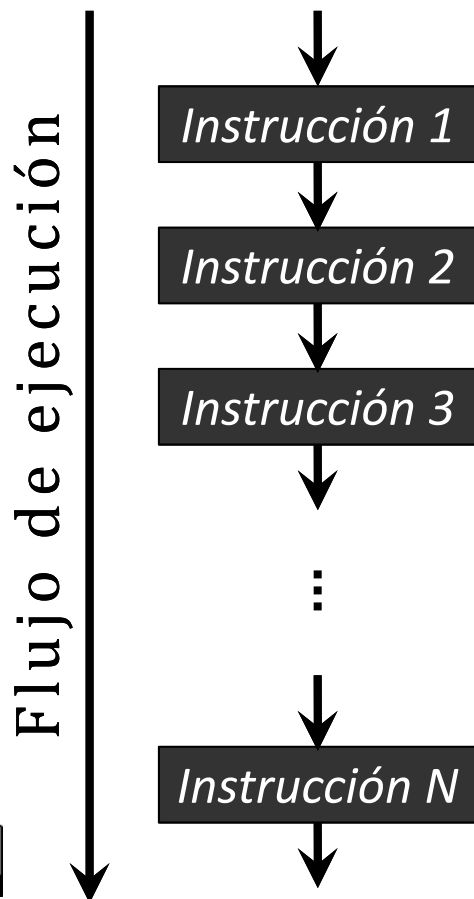
- **Control de flujo:** Orden en que se ejecutan las instrucciones que tenemos en el programa
 - El orden puede ser secuencial, selectivo o repetitivo
- *Flujo secuencial:* Se ejecuta una instrucción detrás de otra en el orden en qué están escritas
- *Flujo selectivo:* Se ejecutan o no bloques de código en función de una condición
- *Flujo repetitivo:* Se repite un bloque de código mientras se cumple una condición



Ejecución secuencial

Por lo general las instrucciones se ejecutan una después de otra, en el orden en que están escritas, es decir, en secuencia

➤ Este proceso se conoce como **ejecución secuencial**



```
double oper1, oper2, prod;  
cout << "Operando 1: ";  
cin >> oper1;  
cout << "Operando 2: ";  
...  
cout << "Producto: " << prod;  
return 0;
```



Transferencia de control (I)

- ❑ Se puede especificar que las siguientes instrucciones a ejecutar no sean las siguientes en secuencia
 - Esto se conoce como **transferencia de control**
- ❑ Las estructuras de control permiten modificar el flujo de ejecución de un programa
- ❑ Con las estructuras de control se puede:
 - De acuerdo a una condición, ejecutar un grupo u otro de instrucciones
 - if
 - De acuerdo al valor de una variable, ejecutar un grupo u otro de instrucciones
 - switch
 - Mientras se cumpla una condición, ejecutar un grupo de instrucciones
 - while
 - Hasta que se cumpla una condición, ejecutar un grupo de instrucciones
 - do-while
 - Ejecutar un grupo de instrucciones un número determinado de veces
 - for



Transferencia de control (II)

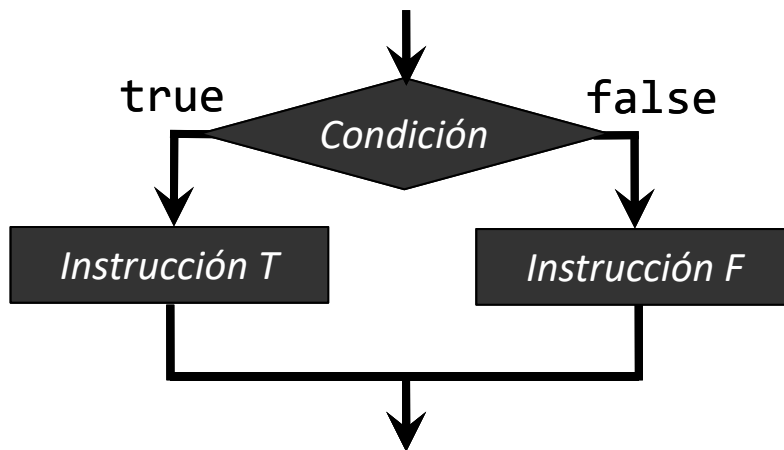
- ❑ Las estructuras de control se puede clasificar en selectivas e iterativas
- ❑ *Estructuras de control selectivas*: ejecutan un bloque de instrucciones u otro según se cumpla o no una condición
 - if, switch
- ❑ *Estructuras de control iterativas*: repiten un bloque de instrucciones si se cumple una condición o mientras se cumple una condición
 - while, do-while, for

Estructuras de control selectivas



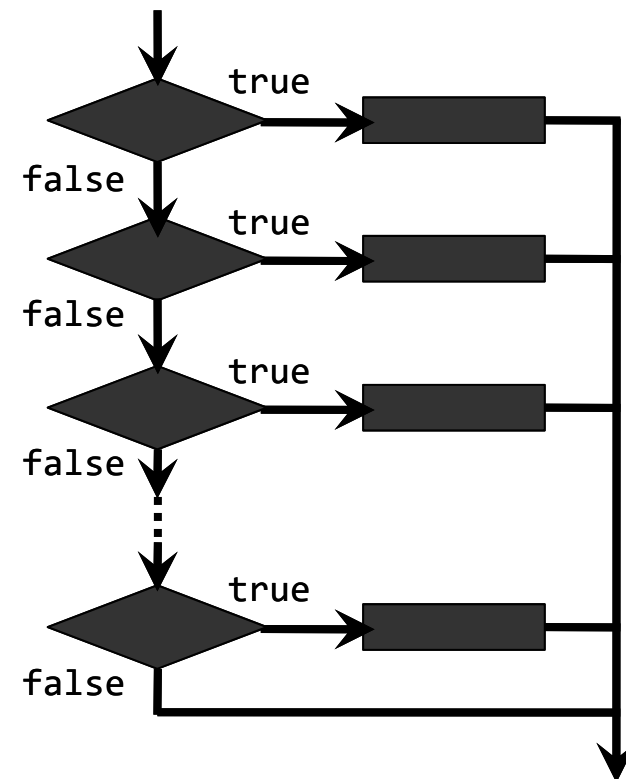
Uno entre dos o más caminos de ejecución

Selección simple (2 caminos)



if

Selección múltiple (> 2 caminos)



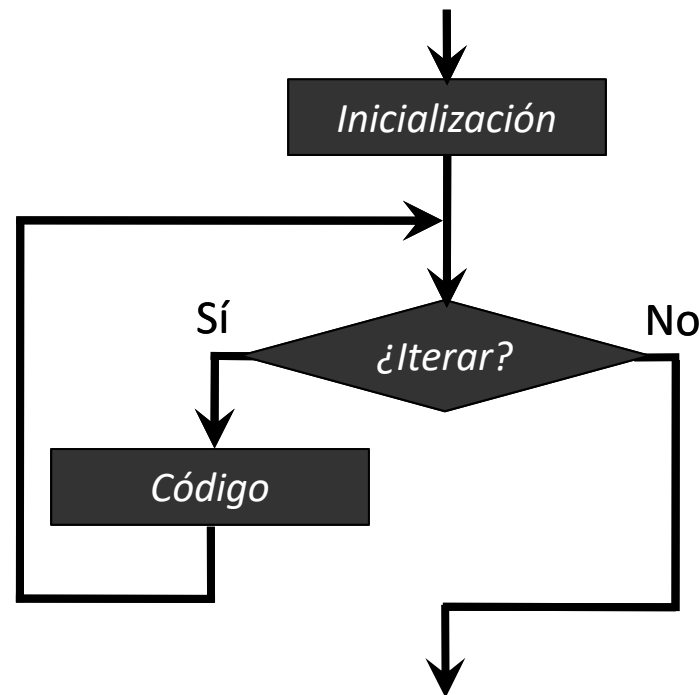
if-else-if
switch



Estructuras de control iterativas



Repetir la ejecución de una o más instrucciones



while
for



Selección simple (bifurcación)

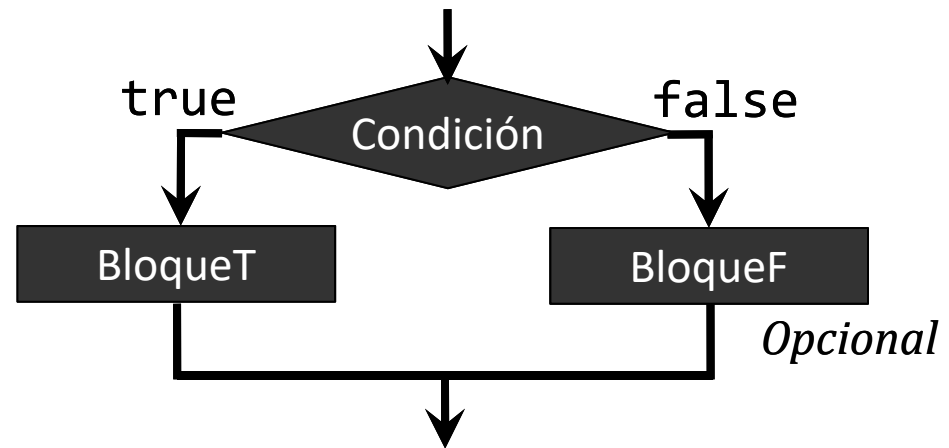


La instrucción if

```
if (condición) {  
    ↪ códigoT  
}  
[else {  
    ↪ códigoF  
}]
```

condición: expresión bool

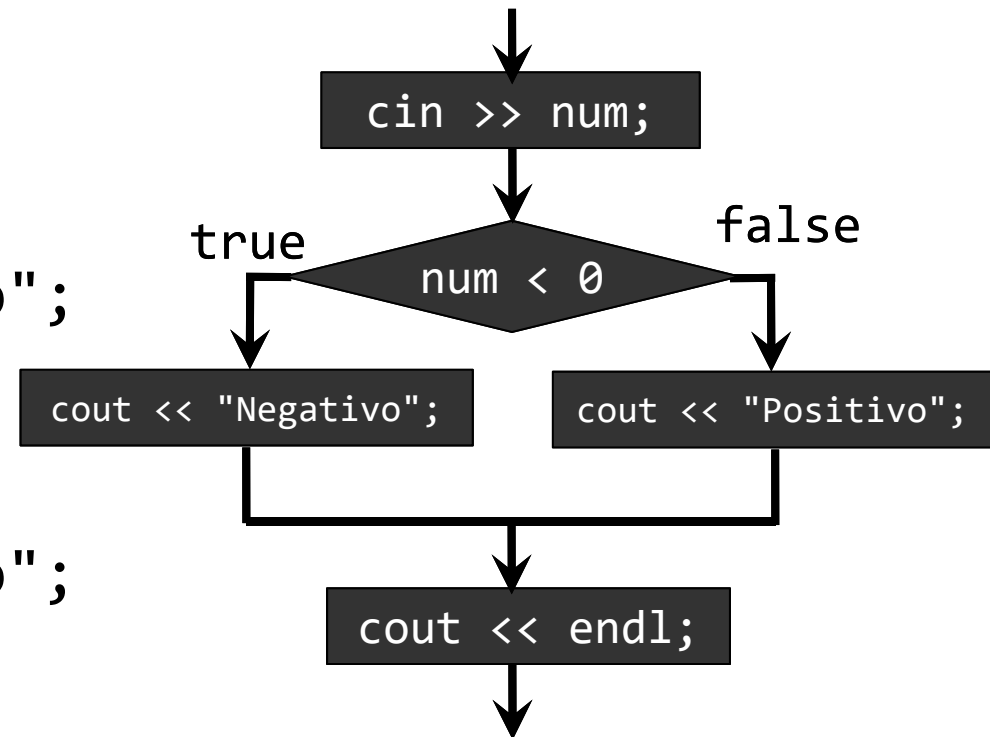
Cláusula else opcional



La instrucción `if`

signo.cpp

```
...  
int num;  
cin >> num;  
if (num < 0) {  
    cout << "Negativo";  
}  
else {  
    cout << "Positivo";  
}  
cout << endl;  
...
```



Bloques de código

Agrupación de instrucciones

Grupo de instrucciones a ejecutar en una rama del if



```
int num, total = 0;
cin >> num;
if (num > 0)
{
    cout << "Positivo";
    total = total + num;
}
cout << endl;
```

```
{
  Tab ó  | instrucción1
3 esp.   | instrucción2
         | ...
         | instrucciónN
}
```



Bloques de código

Posición de las llaves: cuestión de estilo

```
if (num > 0)
{
    cout << "Positivo";
    total = total + num;
}
cout << endl;
```

```
if (num > 0) {
    cout << "Positivo";
    total = total + num;
}
cout << endl;
```

No necesitamos las llaves si sólo hay una instrucción

```
if (num > 0) {
    cout << "Positivo";
}
≡
if (num > 0)
    cout << "Positivo";
```

Evita poner el if y la instrucción objetivo en la misma línea:

```
if (num > 0) cout << "Positivo";
```



Selección simple. Ejemplo

división.cpp

División entre dos números protegida frente a intento de división por 0

```
#include <iostream>
using namespace std;

int main() {
    double numerador, denominador, resultado;
    cout << "Numerador: ";
    cin >> numerador;
    cout << "Denominador: ";
    cin >> denominador;
    if (denominador == 0) {
        cout << "Imposible dividir entre 0!";
    }
    else {
        resultado = numerador / denominador;
        cout << "Resultado: " << resultado << endl;
    }
    return 0;
}
```



Operadores lógicos (booleanos)

Se aplican a valores `bool` (*condiciones*)

El resultado es de tipo `bool`

!	NO	Monario
&&	Y	Binario
	O	Binario

Operadores (prioridad)
...
!
* / %
+ -
< <= > >=
== !=
&&



Operadores lógicos - Tablas de verdad

!		&&		true	false			true	false
true	false	true	true	true	false	true	true	true	true
false	true	false	false	false	false	false	true	false	false

NO (*Not*)

Y (*And*)

O (*Or*)

```
bool cond1, cond2, resultado;
int a = 2, b = 3, c = 4;
resultado = !(a < 5);           // !(2 < 5) → !true → false
cond1 = (a * b + c) >= 12;      // 10 >= 12 → false
cond2 = (a * (b + c)) >= 12;    // 14 >= 12 → true
resultado = cond1 && cond2;      // false && true → false
resultado = cond1 || cond2;     // false || true → true
```



Condiciones

- Condición simple: Expresión lógica (true/false)
Sin operadores lógicos

`num < 0`

`car == 'a'`

`isalpha(car)`

`12`

0 es equivalente a false

Cualquier valor distinto de 0 es equivalente a true

- Condición compuesta:
Combinación de condiciones simples y operadores lógicos

`!isalpha(car)`

`(num < 0) || (car == 'a')`

`(num < 0) && ((car == 'a') || !isalpha(car))`



No confundas el operador de igualdad (==)
con el operador de asignación (=).



Ejemplo

condiciones.cpp

```
#include <iostream>
using namespace std;

int main()
{
    int num;
    cout << "Introduce un número entre 1 y 10: ";
    cin >> num;
    if ((num >= 1) && (num <= 10)) {
        cout << "Número dentro del intervalo de valores válidos";
    }
    else {
        cout << "Número no válido!";
    }
    return 0;
}
```



¡Encierra las condiciones simples entre paréntesis!

Condiciones equivalentes

```
((num >= 1) && (num <= 10))
((num > 0) && (num < 11))
((num >= 1) && (num < 11))
((num > 0) && (num <= 10))
```



Anidamiento de if's

diasmes.cpp

Número de días de un mes

```
int mes, anio, dias;
cout << "Número de mes: ";
cin >> mes;
cout << "Año: ";
cin >> anio;
if (mes == 2) {
    if (bisiesto(mes, anio)) { // bisiesto es una función que devuelve un bool
        dias = 29;
    }
    else {
        dias = 28;
    }
}
else {
    if ((mes == 1) || (mes == 3) || (mes == 5) || (mes == 7)
        || (mes == 8) || (mes == 10) || (mes == 12)) {
        dias = 31;
    }
    else {
        dias = 30;
    }
}
```



¿Año bisiesto?

Calendario Gregoriano: bisiesto si divisible por 4, excepto el último de cada siglo (divisible por 100), salvo que sea divisible por 400

```
bool bisiesto(int mes, int anio) {  
    bool esBisiesto;  
    if ((anio % 4) == 0) { // Divisible por 4  
        if (((anio % 100) == 0) && ((anio % 400) != 0)) {  
            // Pero último de siglo y no múltiplo de 400  
            esBisiesto = false;  
        }  
        else {  
            esBisiesto = true; // Año bisiesto  
        }  
    }  
    else {  
        esBisiesto = false;  
    }  
    return esBisiesto;  
}
```



Asociación de cláusulas `else`

Cada `else` se asocia al `if` anterior más cercano sin asociar (mismo bloque)

```
if (condición1) {  
    if (condición2) {...}  
    else {...}  
}  
else {  
    if (condición3) {  
        if (condición4) {...}  
        if (condición5) {...}  
        else {...}  
    }  
    else { ...  
}
```

Una mala sangría puede confundir

```
if (x > 0) {  
    if (y > 0) {...}  
    else {...}
```



```
if (x > 0) {  
    if (y > 0) {...}  
    else {...}
```



La sangría ayuda a asociar los `else` con sus `if`



Ejercicio

Escribe un programa en C++ que pida al usuario tres valores enteros y los muestre de menor a mayor separados por comas. Por ejemplo, si el usuario introduce 10, 4 y 6, el resultado será: 4, 6, 10.

¿Cómo lo resolveríais?

Ejercicio. Solución. 15 min

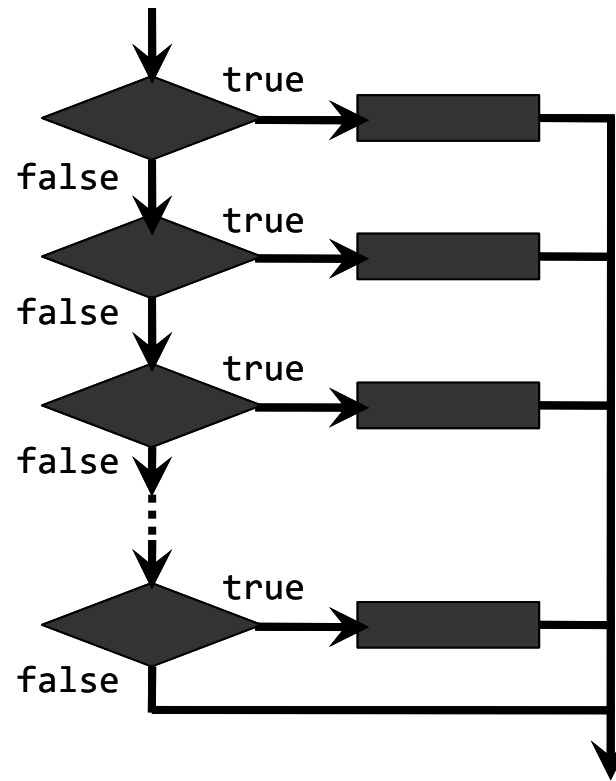
```
#include <iostream>
using namespace std;

int main()
{
    int num1, num2, num3, tmp;
    cout << "Primer número: ";
    cin >> num1;
    cout << "Segundo número: ";
    cin >> num2;
    cout << "Tercer número: ";
    cin >> num3;
    if (num1 > num2) {
        // Si el primero es mayor que el segundo, intercambiamos
        tmp = num1;
        num1 = num2;
        num2 = tmp;
    }
    if (num1 > num3) {
        // Si el primero es mayor que el tercero, intercambiamos
        tmp = num1;
        num1 = num3;
        num3 = tmp;
    }
    if (num2 > num3) {
        // Si el segundo es mayor que el tercero, intercambiamos
        tmp = num2;
        num2 = num3;
        num3 = tmp;
    }
    cout << num1 << "," << num2 << "," << num3 << endl;

    return 0;
}
```



Selección múltiple



if-else-if

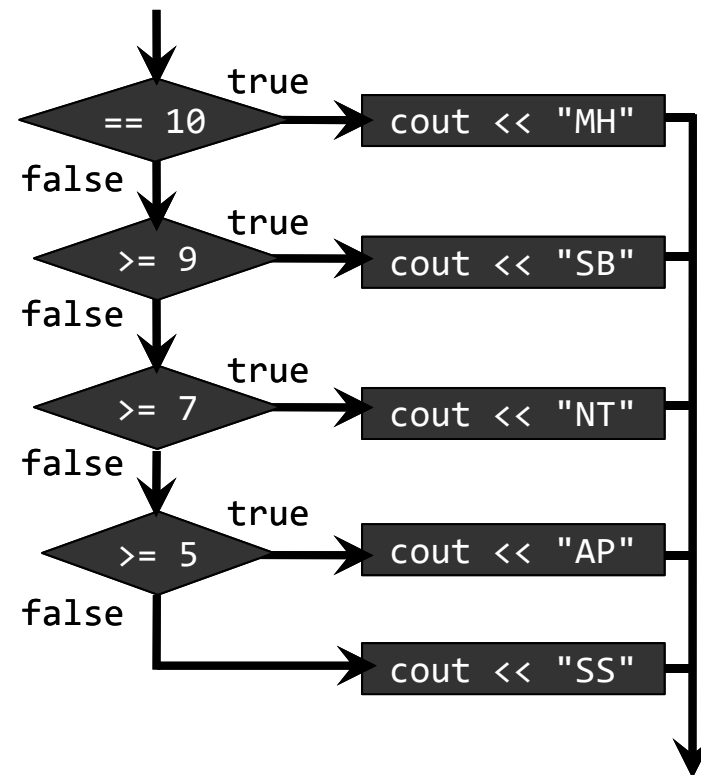


La escala if-else-if (I)

Ejemplo:

Calificación (en letras)
de un estudiante en base
a su nota numérica (0-10)

Si nota == 10 entonces MH
si no, si nota >= 9 entonces SB
si no, si nota >= 7 entonces NT
si no, si nota >= 5 entonces AP
si no SS



La escala if-else-if (II)

nota.cpp

```
double nota;  
cin >> nota;  
if (nota == 10) {  
    cout << "MH";  
}  
else {  
    if (nota >= 9) {  
        cout << "SB";  
    }  
    else {  
        if (nota >= 7) {  
            cout << "NT";  
        }  
        else {  
            if (nota >= 5) {  
                cout << "AP";  
            }  
            else {  
                cout << "SS";  
            }  
        }  
    }  
}
```

≡

```
double nota;  
cin >> nota;  
if (nota == 10) {  
    cout << "MH";  
}  
else if (nota >= 9) {  
    cout << "SB";  
}  
else if (nota >= 7) {  
    cout << "NT";  
}  
else if (nota >= 5) {  
    cout << "AP";  
}  
else {  
    cout << "SS";  
}
```



La escala if-else-if (III)

¡Cuidado con el orden de las condiciones!

```
double nota;  
cin >> nota;  
if (nota < 5) { cout << "SS"; }  
else if (nota < 7) { cout << "AP"; }  
else if (nota < 9) { cout << "NT"; }  
else if (nota < 10) { cout << "SB"; }  
else { cout << "MH"; }
```



```
double nota;  
cin >> nota;  
if (nota >= 5) { cout << "AP"; }  
else if (nota >= 7) { cout << "NT"; }  
else if (nota >= 9) { cout << "SB"; }  
else if (nota == 10) { cout << "MH"; }  
else { cout << "SS"; }
```

¡No se ejecutan nunca!



Sólo muestra AP o SS



La escala if-else-if (IV)

Simplificación de las condiciones



```
if (nota == 10) { cout << "MH"; }
else if ((nota < 10) && (nota >= 9)) { cout << "SB"; }
else if ((nota < 9) && (nota >= 7)) { cout << "NT"; }
else if ((nota < 7) && (nota >= 5)) { cout << "AP"; }
else if (nota < 5) { cout << "SS"; }
```

```
if (nota == 10) { cout << "MH"; }
else if (nota >= 9) { cout << "SB"; }
else if (nota >= 7) { cout << "NT"; }
else if (nota >= 5) { cout << "AP"; }
else { cout << "SS"; }
```

Siempre true: ramas else
Si no es 10, es menor que 10
Si no es ≥ 9 , es menor que 9
Si no es ≥ 7 , es menor que 7
...
 $\text{true} \ \&\& \ X \equiv X$



Nivel de un valor (I)

nivel.cpp

```
#include <iostream>
using namespace std;
int main() {
    int num;
    cout << "Introduce el nivel: ";
    cin >> num;

    if (num == 4)
        cout << "Muy alto" << endl;
    else
        if (num == 3)
            cout << "Alto" << endl;
        else
            if (num == 2)
                cout << "Medio" << endl;
            else
                if (num == 1)
                    cout << "Bajo" << endl;
                else
                    cout << "Valor no válido" << endl;

    return 0;
}
```

Si num == 4 entonces Muy alto
Si num == 3 entonces Alto
Si num == 2 entonces Medio
Si num == 1 entonces Bajo



Nivel de un valor (II)

```
#include <iostream>
using namespace std;

int main() {
    int num;
    cout << "Introduce el nivel: ";
    cin >> num;
    if (num == 4)
        cout << "Muy alto" << endl;
    else if (num == 3)
        cout << "Alto" << endl;
    else if (num == 2)
        cout << "Medio" << endl;
    else if (num == 1)
        cout << "Bajo" << endl;
    else
        cout << "Valor no válido" << endl;
    return 0;
}
```



¿Código repetido en las distintas ramas?

```
if (num == 4) { cout << "Muy alto" << endl; }  
else if (num == 3) { cout << "Alto" << endl; }  
else if (num == 2) { cout << "Medio" << endl; }  
else if (num == 1) { cout << "Bajo" << endl; }  
else cout << "Valor no válido" << endl; }
```



```
if (num == 4) cout << "Muy alto";  
else if (num == 3) cout << "Alto";  
else if (num == 2) cout << "Medio";  
else if (num == 1) cout << "Bajo";  
else cout << "Valor no válido";  
cout << endl;
```



Ejercicio 15 min

Debido a una pertinaz sequía se decidió poner en práctica un sistema de cobro de agua que penalice el consumo excesivo tal como indica la tabla siguiente:

Consumo (m ³)	€/ m ³
Primeros 100	0,15
De 100 a 500	0,20
De 500 a 1000	0,35
Más de 1000	0,80

Escribe un programa que lea del teclado los metros cúbicos consumidos y muestre en la pantalla el coste de agua total. Ten en cuenta que en la tabla se indica lo que hay que cobrar por los m³ que se encuentran en el intervalo. Así, si hemos consumido 750 m³ deberíamos pagar:

$$100 * 0,15 + 400 * 0,20 + 250 * 0,35 = 182,50 \text{ €}.$$

Ejercicio. Solución 1

```
#include <iostream>
using namespace std;

int getCoste(double cantidad);

int main () {
    double cantidad;

    cout << "Introduce los metros cubicos de agua consumidos: ";
    cin >> cantidad;
    if (cantidad < 0)
        cout << "¡Ha de ser positiva!" << endl;
    else
        cout << "Coste: " << getCoste(cantidad) << endl;

    return 0;
}

double getCoste(double cantidad){
    double coste;
    const int Limite1 = 100, Limite2 = 500, Limite3 = 1000;
    const double Tarifa1 = 0.15, Tarifa2 = 0.2, Tarifa3 = 0.35, Tarifa4 = 0.8;
    if (cantidad <= Limite1)
        coste = cantidad * Tarifa1;
    else if (cantidad <= Limite2)
        coste = Limite1 * Tarifa1 + (cantidad - Limite1) * Tarifa2;
    else if (cantidad <= Limite3)
        coste = Limite1 * Tarifa1 + (Limite2 - Limite1) * Tarifa2 + (cantidad - Limite2) * Tarifa3;
    else
        coste = Limite1 * Tarifa1 + (Limite2 - Limite1) * Tarifa2 + (Limite3 - Limite2) * Tarifa3 + (cantidad - Limite3) * Tarifa4;

    return coste;
}
```


Ejercicio. Solución 2

```
#include <iostream>
using namespace std;

int getCoste(double cantidad);

int main () {
    double cantidad;

    cout << "Introduce los metros cubicos de agua consumidos: ";
    cin >> cantidad;
    if (cantidad < 0)
        cout << "¡Ha de ser positiva!" << endl;
    else
        cout << "Coste: " << getCoste(cantidad) << endl;

    return 0;
}

double getCoste(double cantidad){
    double coste;
    int double Limite1 = 100, Limite2 = 500, Limite3 = 1000;
    const double Tarifa1 = 0.15, Tarifa2 = 0.2, Tarifa3 = 0.35, Tarifa4 = 0.8;

    coste = 0;
    if (cantidad > Limite3) {
        coste = coste + (cantidad - Limite3) * Tarifa4;
        cantidad = Limite3;
    }
    if (cantidad > Limite2) {
        coste = coste + (cantidad - Limite2) * Tarifa3;
        cantidad = Limite2;
    }
    if (cantidad > Limite1) {
        coste = coste + (cantidad - Limite1) * Tarifa2;
        cantidad = Limite1;
    }
    if (cantidad > 0) {
        coste = coste + cantidad * Tarifa1;
    }

    return coste;
}
```



La instrucción switch

Selección entre valores posibles de una expresión

```
switch (expresión) {  
  case constante1:  
    {  
      código1  
    }  
    [break;]  
  case constante2:  
    {  
      código2  
    }  
    [break;]  
  ...  
}
```

→

```
case constanteN:  
{  
  códigoN  
}  
[break;]  
[default:  
{  
  códigoDefault  
}]  
}
```



La instrucción switch

nivel2.cpp

```
switch (num) {  
  case 4:  
  {  
    cout << "Muy alto";  
  }  
  break;  
  case 3:  
  {  
    cout << "Alto";  
  }  
  break;  
  case 2:  
  {  
    cout << "Medio";  
  }  
  break;  
  case 1:  
  {  
    cout << "Bajo";  
  }  
  break;  
  default:  
  {  
    cout << "Valor no válido";  
  }  
}
```

Si num == 4 → Muy alto

Si num == 3 → Alto

Si num == 2 → Medio

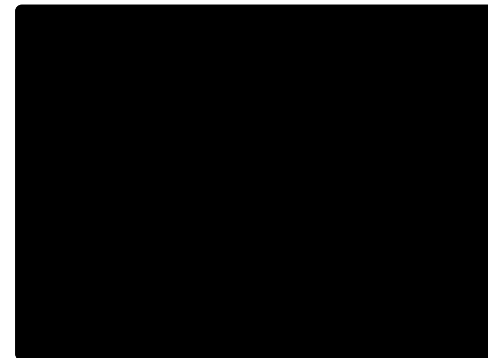
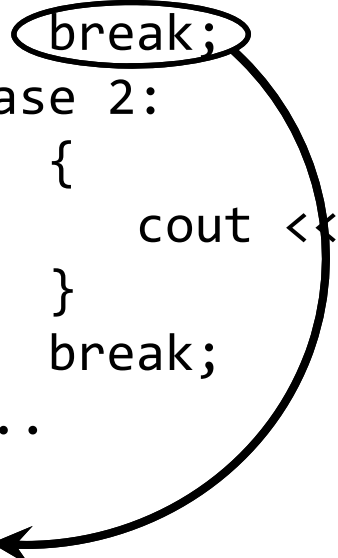
Si num == 1 → Bajo



La instrucción break (I)

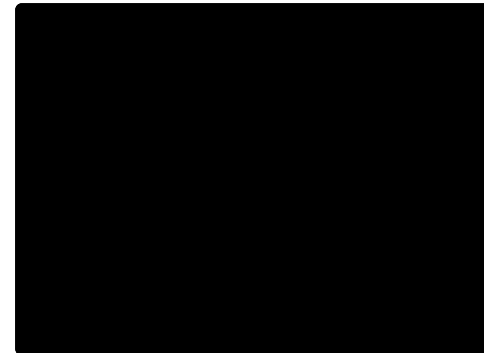

Interrumpe el switch

```
switch (num) {  
    ...  
    case ③:  
        {  
            cout << "Alto";  
        }  
        break;  
    case 2:  
        {  
            cout << "Medio";  
        }  
        break;  
    ...  
}
```

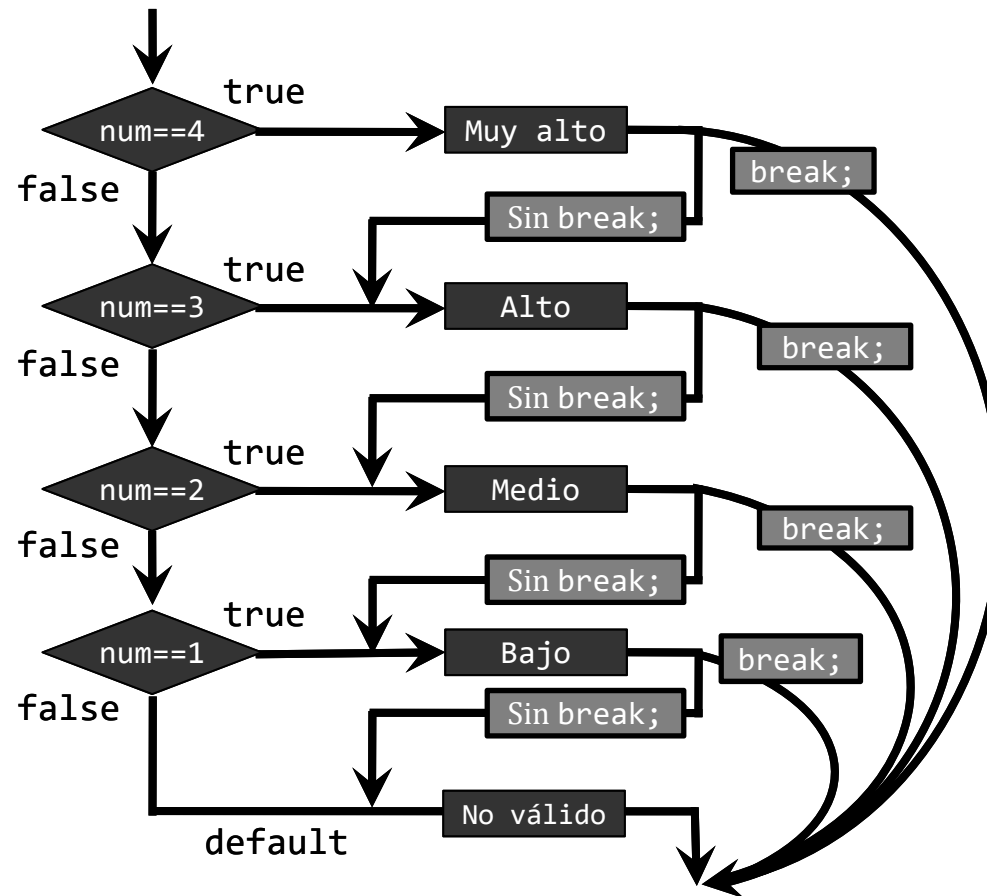


La instrucción break (II)

```
switch (num) {  
    ...  
    case ③:  
    {  
        cout << "Alto";  
    }  
    case 2:  
    {  
        cout << "Medio";  
    }  
    case 1:  
    {  
        cout << "Bajo";  
    }  
    default:  
    {  
        cout << "Valor no válido";  
    }  
}
```



Con y sin break



Casos múltiples

nota2.cpp

```
int nota; // Sin decimales
cout << "Nota (0-10): ";
cin >> nota;
switch (nota) {
case 0:
case 1:
case 2:
case 3:
case 4:
    {
        cout << "Suspenso";
    }
    break; // De 0 a 4: SS
case 5:
case 6:
    {
        cout << "Aprobado";
    }
    break; // 5 o 6: AP
```

```
case 7:
case 8:
    {
        cout << "Notable";
    }
    break; // 7 u 8: NT
case 9:
case 10:
    {
        cout << "Sobresaliente";
    }
    break; // 9 o 10: SB
default:
    {
        cout << "¡No válida!";
    }
}
```



Ejercicio

*Escribe un programa en C++ que lea un operando (real), un operador (carácter) y otro operando (real), todo en una misma línea, y muestre el resultado de la operación correspondiente (operadores contemplados: +, -, * y /).*

```
D:\FP\Tema 2>02-19
Operando Operador Operando (0 para terminar): 12 + 4
12 + 4 = 16
```

Emplea una función para calcular el resultado de la operación.

Ejercicio. Solución

```
#include <iostream>
using namespace std;

double getResultado(double op1, double op2, char operador);

int main()
{
    double op1, op2;
    char operador;
    |
    cout << "Operando Operador Operando: ";
    cin >> op1;
    cin >> operador >> op2;

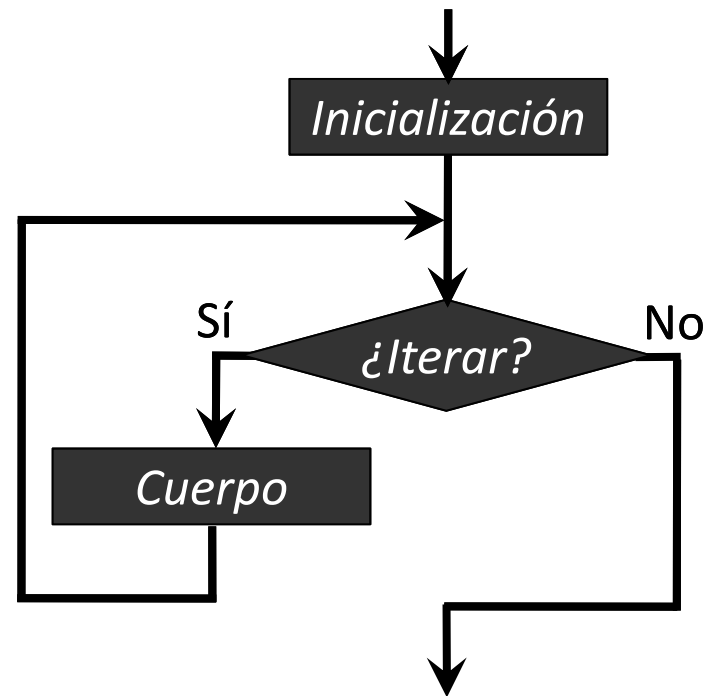
    cout << op1 << " " << operador << " " << op2
        << " = " << getResultado(op1, op2, operador) << endl;

    return 0;
}

double getResultado(double op1, double op2, char operador)
{
    double resultado;
    switch(operador){
        case '+':
            resultado = op1 + op2;
            break;
        case '-':
            resultado = op1 - op2;
            break;
        case '*':
            resultado = op1 * op2;
            break;
        case '/':
            resultado = op1 / op2;
            break;
    }
    return resultado;
}
```



Repetición (iteración)



Bucles while y for



Tipos de bucles

✓ Número de iteraciones condicionado (*recorrido variable*):

— Bucle while

`while (condición) cuerpo`

Ejecuta el *cuerpo* mientras la *condición* sea true

— Bucle do-while

`do cuerpo while (condición)`

Comprueba la condición al final, el cuerpo se ejecuta al menos una vez

✓ Número de iteraciones prefijado (*recorrido fijo*):

— Bucle for

`for (inicialización; condición; paso) cuerpo`

Ejecuta el *cuerpo* mientras la *condición* sea true

Se usa una variable contadora entera



El bucle while (I)

while.cpp

Mientras la condición sea cierta, ejecuta el cuerpo

```
while (condición) {  
    cuerpo  
}
```

Condición al principio del bucle

```
int i = 1; // Inicialización de la variable i  
while (i <= 100) {  
    cout << i << endl;  
    i++;  
}
```

Muestra los números del 1 al 100



El bucle while (II)

¿Y si la condición es falsa al comenzar?

No se ejecuta el cuerpo del bucle ninguna vez

```
int op;  
cout << "Introduce la opción: ";  
cin >> op;  
while ((op < 0) || (op > 4)) {  
    cout << "¡No válida! Inténtalo otra vez" << endl;  
    cout << "Introduce la opción: ";  
    cin >> op;  
}
```

Si el usuario introduce un número entre 0 y 4:

No se ejecuta el cuerpo del bucle



Ejemplo de bucle while

primero.cpp

Primer entero cuyo cuadrado es mayor que 1.000

```
#include <iostream>
using namespace std;
```

*¡Ejecuta el programa para
saber cuál es ese número!*

```
int main() {
    int num = 1;
```

← Empezamos en 1

```
    while (num * num <= 1000) {
        num++;
    }
```

← Incrementamos en 1

```
    cout << "1er. entero con cuadrado mayor que 1.000: "
         << num << endl;
```

```
    return 0;
}
```

Recorre la *secuencia* de números 1, 2, 3, 4, 5, ...



Suma y media de números

sumamedia.cpp

```
#include <iostream>
using namespace std;
int main() {
    double num, suma = 0, media = 0;
    int cont = 0;
    cout << "Introduce un número (0 para terminar): ";
    cin >> num;
    while (num != 0) { // 0 para terminar
        suma = suma + num;
        cont++;
        cout << "Introduce un número (0 para terminar): ";
        cin >> num;
    }
    if (cont > 0) {
        media = suma / cont;
    }
    cout << "Suma = " << suma << endl;
    cout << "Media = " << media << endl;
    return 0;
}
```

← Leemos el primero

← Leemos el siguiente



Ejercicio

Implementa un programa que calcule el primer número natural cuyo cubo supera estrictamente otro entero N dado ($N \geq 0$). El programa mostrará la secuencia de números recorrida.



Ejercicio. Solución

```
#include <iostream>
using namespace std;
#include <cmath>

int main() {
    int numero, N;

    cout << "Introduce el valor N a superar: ";
    cin >> N;

    numero = 1;
    // Mientras el cubo no supere N, generamos elementos de la secuencia
    while (pow(double(numero), 3) <= N) { // pow() requiere que el primer argumento sea double
        cout << numero << " ---> " << pow(double(numero), 3) << endl; // Traza
        numero++; // Siguiendo
    }

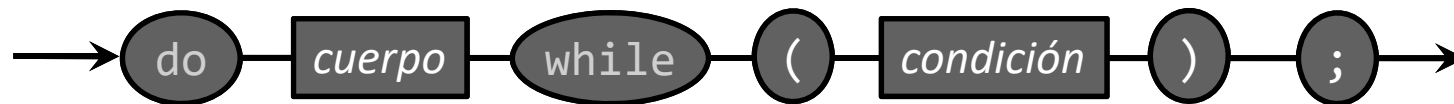
    cout << "Primer natural cuyo cubo supera estrictamente " << N << ": "
        << numero << endl;

    return 0;
}
```



El bucle do..while

`do cuerpo while (condición);` Condición al final del bucle



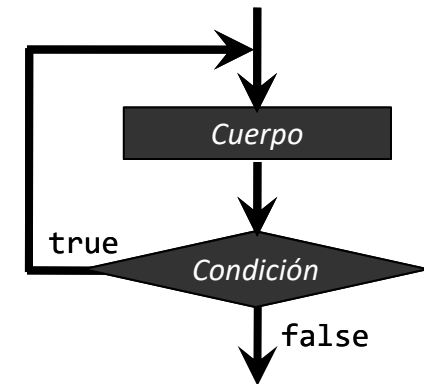
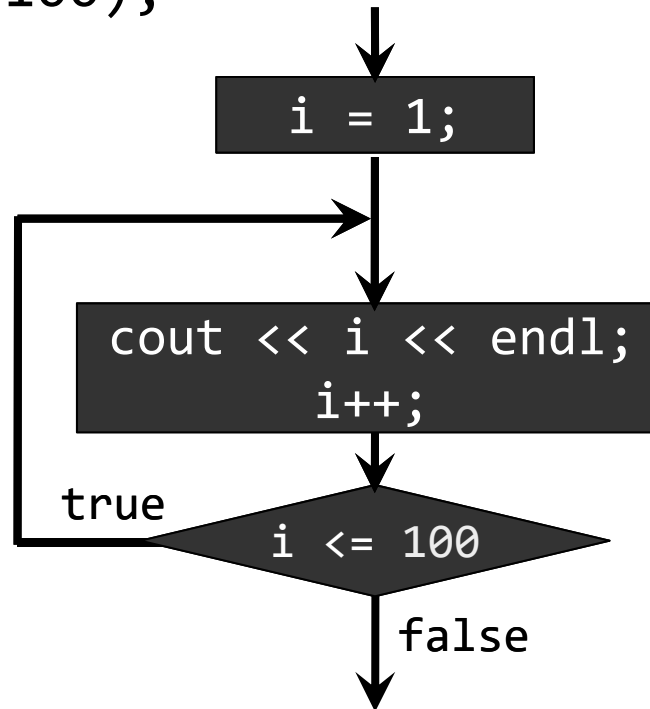
```
int i = 1;
do {
    cout << i << endl;
    i++;
} while (i <= 100);
```

- ✓ El *cuerpo* siempre se ejecuta al menos una vez
- ✓ El *cuerpo* es un bloque de código



Ejecución del bucle do-while

```
int i = 1;
do {
    cout << i << endl;
    i++;
} while (i <= 100);
```



El cuerpo
se ejecuta
al menos
una vez



while versus do-while

¿Ha de ejecutarse al menos una vez el cuerpo del bucle?

```
cin >> d; // Lectura del 1º
while (d != 0) {
    suma = suma + d;
    cont++;
    cin >> d;
}
```

```
do {
    cin >> d;
    if (d != 0) { // ¿Final?
        suma = suma + d;
        cont++;
    }
} while (d != 0);
```

```
cout << "Opción: ";
cin >> op; // Lectura del 1º
while ((op < 0) || (op > 4)) {
    cout << "Opción: ";
    cin >> op;
}
```

```
do { // Más simple
    cout << "Opción: ";
    cin >> op;
} while ((op < 0) || (op > 4));
```



Ejercicio. Tema 4

Escribe un programa que solicite al usuario un número entero positivo del teclado y muestre la suma de sus dígitos.

Por ejemplo, si el entero es 932, mostrará 14 ($9 + 3 + 2$).

El programa no parara de solicitar el número al usuario hasta que el número introducido sea positivo.

El programa usará una función que calcule la suma de los dígitos de un entero.

Ejercicio. Solución

```
#include <iostream>
using namespace std;

int sumaDigitos(int n);

int main() {
    int num;

    // Pedimos el número y comprobamos que sea positivo
    do {
        cout << "Introduce un número positivo: ";
        cin >> num;
    } while (num <= 0);

    cout << "La suma de los dígitos es: " << sumaDigitos(num) << endl;

    return 0;
}

int sumaDigitos(int n) {
    int suma = 0;

    // Acumulo el último dígito con el modulo
    // Elimino el último dígito con la división entera
    while (n > 0) {
        suma = suma + (n % 10);
        n = n / 10;
    }

    return suma;
}
```



Bucle for

Número de iteraciones prefijado

`for ([int] var = ini; condición; paso) cuerpo`

- ✓ Variable contadora que determina el número de iteraciones
- ✓ La *condición* compara el valor de *var* con un valor final
- ✓ El *paso* incrementa o decrementa el valor de *var*
- ✓ El valor de *var* debe ir aproximándose al valor final

`for (int i = 1; i <= 100; i++)...` 1, 2, 3, 4, 5, ..., 100

`for (int i = 100; i >= 1; i--)...` 100, 99, 98, 97, ..., 1

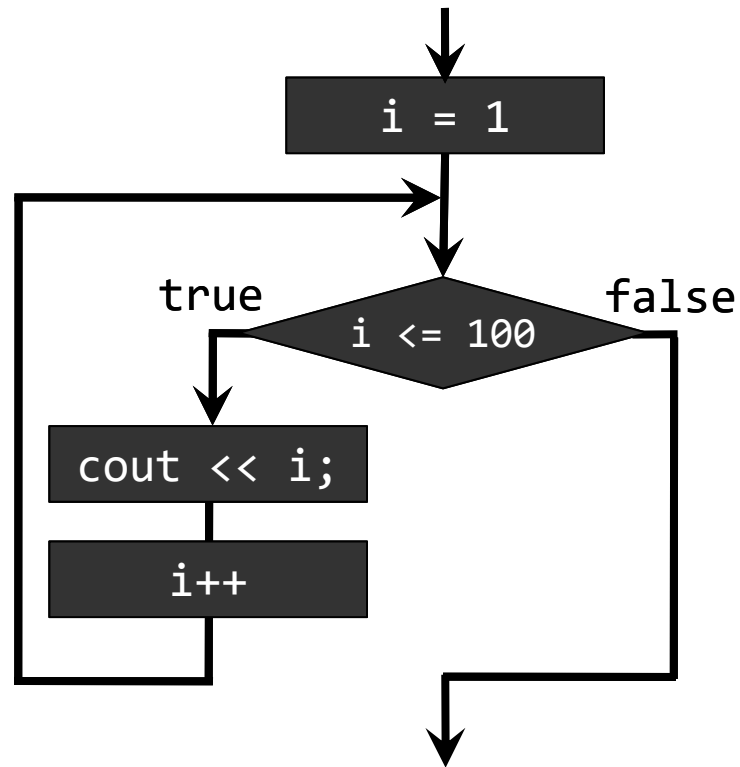
- ✓ Tantos ciclos como valores toma la variable contadora



Ejecución del bucle for (I)

for (*inicialización; condición; paso*) *cuerpo*

```
for (int i = 1; i <= 100; i++) {  
    cout << i;  
}
```



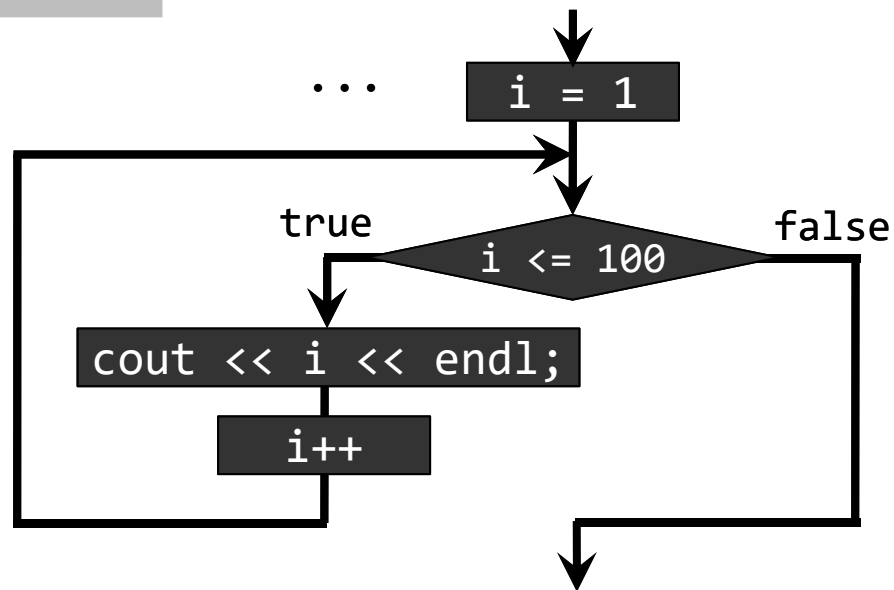
Ejecución del bucle for (II)

for1.cpp

```
for (int i = 1; i <= 100; i++) {  
    cout << i << endl;  
}
```

i

101



1
2
3

99
100



Bucle for . La variable contadora

for2.cpp

El *paso* no tiene porqué ir de uno en uno:

```
for (int i = 1; i <= 100; i = i + 2)
    cout << i << endl;
```

Este bucle for muestra los números impares de 1 a 99



Muy importante

El cuerpo del bucle **NUNCA** debe alterar el valor del contador



Ejemplo de bucle for

suma.cpp

```
#include <iostream>
using namespace std;

long long int suma(int n);

int main() {
    int num;
    cout << "Número final: ";
    cin >> num;
    if (num > 0) { // El número debe ser positivo
        cout << "La suma de los números entre 1 y "
              << num << " es: " << suma(num);
    }
    return 0;
}

long long int suma(int n) {
    long long int total = 0;
    for (int i = 1; i <= n; i++) {
        total = total + i;
    }
    return total;
}
```

$$\sum_{i=1}^N i$$

Recorre la *secuencia* de números
1, 2, 3, 4, 5, ..., n



Bucle for

¿Incremento/decremento prefijo o postfijo?

Es indiferente

Estos dos bucles producen el mismo resultado:

```
for (int i = 1; i <= 100; i++) ...
```

```
for (int i = 1; i <= 100; ++i) ...
```

Bucles infinitos

```
for (int i = 1; i <= 100; i--) ...
```

1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 ...

Cada vez más lejos del valor final (100)

Es un error de diseño/programación

Garantía de terminación

Todo bucle debe terminar su ejecución

Bucles for: la variable contadora debe converger al valor final



Ámbito de la variable contadora

Declarada en el propio bucle

```
for (int i = 1; ...)
```

Sólo se conoce en el cuerpo del bucle (su ámbito)

No se puede usar en instrucciones fuera del bucle

Declarada antes del bucle

```
int i;
```

```
for (i = 1; ...)
```

Se conoce en el cuerpo del bucle y después del mismo

Ámbito externo al bucle



Ejercicio

Escribe un programa en C++ que muestre en la pantalla la tabla de multiplicación (de 1 a 10) del número que introduzca el usuario (entre 1 y 100; si no está en ese intervalo volverá a pedir el número).

La salida debe estar bien formateada, como en este ejemplo:

```
Introduce un numero: 0
Introduce un numero: 212
Introduce un numero: 27
  1 x 27 = 27
  2 x 27 = 54
  3 x 27 = 81
  4 x 27 = 108
  5 x 27 = 135
  6 x 27 = 162
  7 x 27 = 189
  8 x 27 = 216
  9 x 27 = 243
 10 x 27 = 270
```



Ejercicio. Solución

```
#include <iostream>
using namespace std;
#include <iomanip>

int main()
{
    int num, i;
    do{
        cout << "Introduce un número: ";
        cin >> num;
    } while ((num<1) || (num>100));

    for (int i=1;i<=10;i++){
        cout << setw(3) << i << " x "
             << setw(4) << num << " = "
             << setw(5) << i * num << endl;
    }
    return 0;
}
```



Bucles for anidados

Un bucle for en el cuerpo de otro bucle for

Cada uno con su propia variable contadora:

```
for (int i = 1; i <= 100; i++) {  
    for (int j = 1; j <= 5; j++) {  
        cuerpo  
    }  
}
```

Para cada valor de *i* el valor de *j* varía entre 1 y 5

j varía más rápido que i

<i>i</i>	<i>j</i>
1	1
1	2
1	3
1	4
1	5
2	1
2	2
2	3
2	4
2	5
3	1

...



Tablas de multiplicación

tablas.cpp

```
#include <iostream>
using namespace std;
#include <iomanip>

int main() {
    for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
            cout << setw(2) << i << " x "
                 << setw(2) << j << " = "
                 << setw(3) << i * j << endl;
        }
    }

    return 0;
}
```



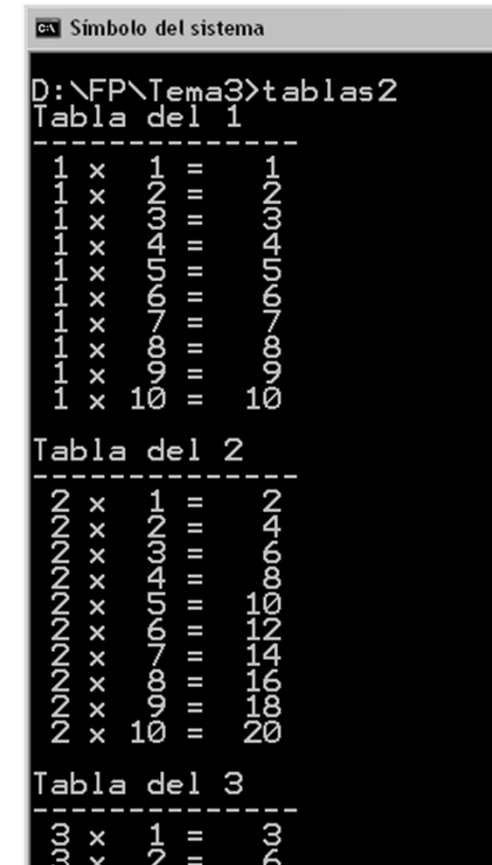
Mejor presentación

tablas2.cpp

```
#include <iostream>
using namespace std;
#include <iomanip>

int main() {
    for (int i = 1; i <= 10; i++) {
        cout << "Tabla del " << i << endl;
        cout << "-----" << endl;
        for (int j = 1; j <= 10; j++) {
            cout << setw(2) << i << " x "
                << setw(2) << j << " = "
                << setw(3) << i * j << endl;
        }
        cout << endl;
    }

    return 0;
}
```



```
Simbolo del sistema
D:\FP\Tema3>tablas2
Tabla del 1
-----
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
Tabla del 2
-----
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
Tabla del 3
-----
3 x 1 = 3
3 x 2 = 6
```



Un menú (I)

```
int menu() {
    int op;

    do {
        cout << "1 - Nuevo cliente" << endl;
        cout << "2 - Editar cliente" << endl;
        cout << "3 - Baja cliente" << endl;
        cout << "4 - Ver cliente" << endl;
        cout << "0 - Salir" << endl;
        cout << "Opción: ";
        cin >> op;

        if ((op < 0) || (op > 4)) {
            cout << "¡Opción no válida!" << endl;
        }
    }while ((op < 0) || (op > 4));

    return op;
}
```



Un menú (II)

```
int opcion;
...
opcion = menu();
switch (opcion) {
case 1:
{
    cout << "En la opción 1..." << endl;
}
break;
case 2:
{
    cout << "En la opción 2..." << endl;
}
break;
case 3:
{
    cout << "En la opción 3..." << endl;
}
break;
case 4:
{
    cout << "En la opción 4..." << endl;
} // En la última no necesitamos break
}
```



El menú con su bucle...

```
int opcion;
...
opcion = menu();
while (opcion != 0) {
    switch (opcion) {
        case 1:
        {
            cout << "En la opción 1..." << endl;
        }
        break;
        case 4:
        {
            cout << "En la opción 4..." << endl;
        }
    } // switch
    ...
    opcion = menu();
} // while
```



Más bucles anidados

menú.cpp

```
#include <iostream>
using namespace std;
#include <iomanip>

int menu(); // 1: Tablas de multiplicación; 2: Sumatorio
long long int suma(int n); // Sumatorio

int main() {
    int opcion = menu();
    while (opcion != 0) {
        switch (opcion) {
            case 1:
                {
                    for (int i = 1; i <= 10; i++) {
                        for (int j = 1; j <= 10; j++) {
                            cout << setw(2) << i << " x "
                                << setw(2) << j << " = "
                                << setw(3) << i * j << endl;
                        }
                    }
                }
            break; ...
        }
    }
}
```



Más bucles anidados

```
case 2:
{
    int num = 0;
    while (num <= 0) {
        cout << "Hasta (positivo)? ";
        cin >> num;
    }
    cout << "La suma de los números del 1 al "
        << num << " es: " << suma(num) << endl;
}
} // switch
opcion = menu();
} // while (opcion != 0)
return 0;
}
```



Más bucles anidados

```
int menu() {
    int op = -1;
    do {
        cout << "1 - Tablas de multiplicar" << endl;
        cout << "2 - Sumatorio" << endl;
        cout << "0 - Salir" << endl;
        cout << "Opción: " << endl;
        cin >> op;
        if ((op < 0) || (op > 2)) {
            cout << "¡Opción no válida!" << endl;
        }
    } while ((op < 0) || (op > 2));
    return op;
}

long long int suma(int n) {
    long long int total = 0;
    for (int i = 1; i <= n; i++) {
        total = total + i;
    }
    return total;
}
```



Ámbito de los identificadores

Cada bloque crea un nuevo ámbito:

```
int main() {  
    double d = -1, suma = 0;           3 ámbitos anidados  
    int cont = 0;  
    while (d != 0) {  
        cin >> d;  
        if (d != 0) {  
            suma = suma + d;  
            cont++;  
        }  
    }  
    cout << "Suma = " << suma << endl;  
    cout << "Media = " << suma / cont << endl;  
    return 0;  
}
```



Ámbito de los identificadores

Un identificador se conoce
en el ámbito en el que está declarado
(a partir de su instrucción de declaración)
y en los subámbitos posteriores



Ámbito de los identificadores

```
int main() {  
    double d;                Ámbito de la variable d  
    if (...) {  
        int cont = 0;  
        for (int i = 0; i <= 10; i++) {  
            ...  
        }  
    }  
    char c;  
    if (...) {  
        double x;  
        ...  
    }  
    return 0;  
}
```



Ámbito de los identificadores

```
int main() {  
    double d;  
    if (...) {  
        int cont = 0;    Ámbito de la variable cont  
        for (int i = 0; i <= 10; i++) {  
            ...  
        }  
    }  
    char c;  
    if (...) {  
        double x;  
        ...  
    }  
    return 0;  
}
```



Ámbito de los identificadores

```
int main() {  
    double d;  
    if (...) {  
        int cont = 0;  
        for (int i = 0; i <= 10; i++) {  
            ...  
        }  
    }  
    char c;  
    if (...) {  
        double x;  
        ...  
    }  
    return 0;  
}
```

Ámbito de la variable i



Ámbito de los identificadores

```
int main() {  
    double d;  
    if (...) {  
        int cont = 0;  
        for (int i = 0; i <= 10; i++) {  
            ...  
        }  
    }  
    char c;  
    if (...) {  
        double x;  
        ...  
    }  
    return 0;  
}
```

Ámbito de la variable c



Ámbito de los identificadores

```
int main() {  
    double d;  
    if (...) {  
        int cont = 0;  
        for (int i = 0; i <= 10; i++) {  
            ...  
        }  
    }  
    char c;  
    if (...) {  
        double x;  
        ...  
    }  
    return 0;  
}
```

Ámbito de la variable x

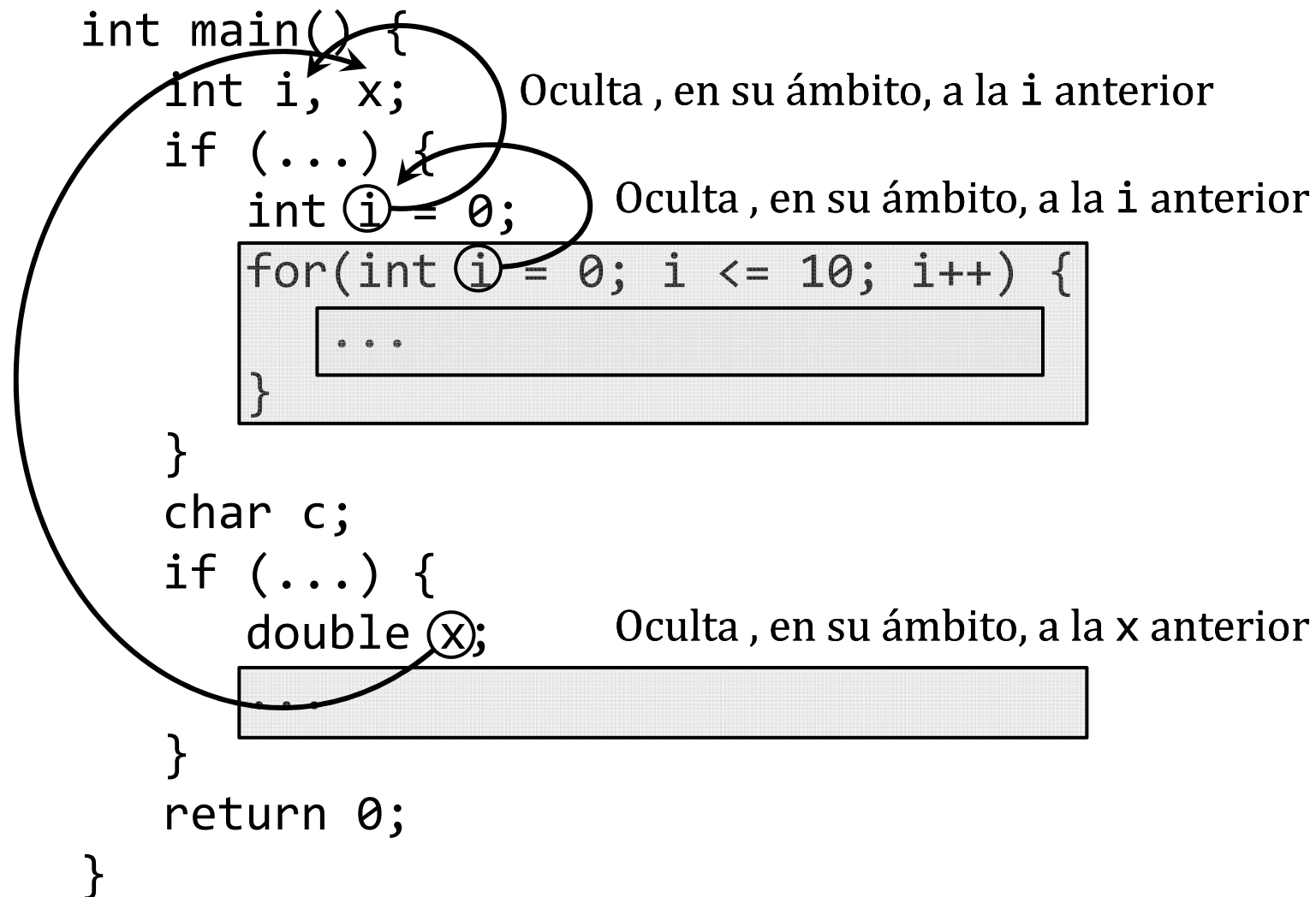


Visibilidad de los identificadores

Si en un subámbito se declara
un identificador con idéntico nombre
que uno ya declarado en el ámbito,
el del subámbito *oculta* al del ámbito
(no es visible)



Visibilidad de los identificadores



Ejercicio bucles anidados (I)

Escribe un programa que calcule los factoriales de los enteros del 1 al 5.

- El factorial de un entero positivo n ($n!$) es igual al producto de los enteros positivos entre 1 y n

```
El factorial de 1 es: 1
El factorial de 2 es: 2
El factorial de 3 es: 6
El factorial de 4 es: 24
El factorial de 5 es: 120
```



Ejercicio bucles anidados (II)

Escribe un programa que pida al usuario un entero positivo y escriba por pantalla todos los números perfectos que hay entre 1 y el número dado.

- Un número se considera perfecto si la suma de todos sus divisores es igual al propio número.

NO se usarán funciones, se hará todo en el programa principal

```
Introduce un entero positivo: 1000
0 es un cuadrado perfecto
6 es un cuadrado perfecto
28 es un cuadrado perfecto
496 es un cuadrado perfecto
```






Acerca de *Creative Commons*



Licencia CC (Creative Commons)

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

-  Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
-  No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.
-  Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Material original elaborado por Luis Hernández Yáñez, con modificaciones de Raquel Hervás, Virginia Francisco y Javier Arroyo.

